Manual for the software EMmod

Jürg Hunziker[1], Jan Thorbecke[2] and Evert Slob[3]
Delft University of Technology, Delft, The Netherlands

Delft University of Technology, Delft, The Netherlands
[1]j.w.hunziker@tudelft.nl
[2]j.w.thorbecke@tudelft.nl
[3]e.c.slob@tudelft.nl

1 Introduction

EMmod, which stands for electromagnetic modeling, is an algorithm to model the full electromagnetic field in a vertical transverse isotropic (VTI) layered medium. Although the algorithm was designed for frequency-domain marine Controlled Source Electromagnetics (CSEM), it can also be used to simulate data for other geophysical exploration methods like for example Ground Penetrating Radar (GPR). The algorithm computes the EM-field analytically in the wavenumber-frequency domain and then carries out a numerical Hankel transformation to get the space-frequency domain representation. The expressions on which the algorithm is based are given in Hunziker et al. (2014). The algorithm is able to simulate diffusive electromagnetic fields as well as electromagnetic waves. The user has complete freedom in placing the source and the receivers, i.e., they can be placed at any depth. The algorithm is also able to compute the reflection response of the subsurface, which can be retrieved by interferometry by multidimensional deconvolution (Hunziker et al., 2013). Thus, the algorithm can be used to produce a sort of "ground truth" for electromagnetic interferometry studies.
The structure of the program is schematically shown in Figure 1. Roughly, it can be said, that the algorithm consists of three major parts:

1. Everything before the while loop
2. Everything in the while loop
3. Everything after the while loop

During the first major part (everything before the while loop) all the parameters are read from the input file and the arrays are allocated. It is further determined which component, i.e., which source and receiver geometry, the user has chosen. Some source-receiver geometries do not feature both, the TM- and the TE-mode, and also the symmetry depends on the source-receiver geometry. Therefore, each source-receiver geometry has its own section in the code in which the corresponding electromagnetic field is computed in the wavenumber domain.
The second part consists of the while loop. In the while loop, the Hankel transformation is carried out for a line of coordinates in the space domain. The integral in the Hankel transformation is computed using the 61-point Gauss-Kronrod rule from the slatec library (Piessens et al., 1983). In the final step of the while loop it is assessed if enough points in the space domain have been computed to build up the complete space-domain grid from that line. This assessment is done by interpolating the electromagnetic field at one point using the two neighboring points. If the interpolated point differs from the integrated electromagnetic field at that location by less than a predefined value, no additional points are required. If the difference is larger, two more points are added. The Hankel transformation of the electromagnetic field at those locations will be computed in the next iteration of the while loop. The while loop ends if all points of the line are within the required precision or if the Hankel transformation has been evaluated for more than a specified amount of points.
In the third major part (everything after the while loop) the electromagnetic field for one quadrant of the grid in the space domain is computed based on the line data from the Hankel transformation and the

corresponding cosine or sine factors. Next, the electromagnetic field for the complete grid is built up by coping and pasting of the quadrant already computed and by taking into account the symmetry of the field. Finally, the electromagnetic field is written to the output file on disk and the program stops.

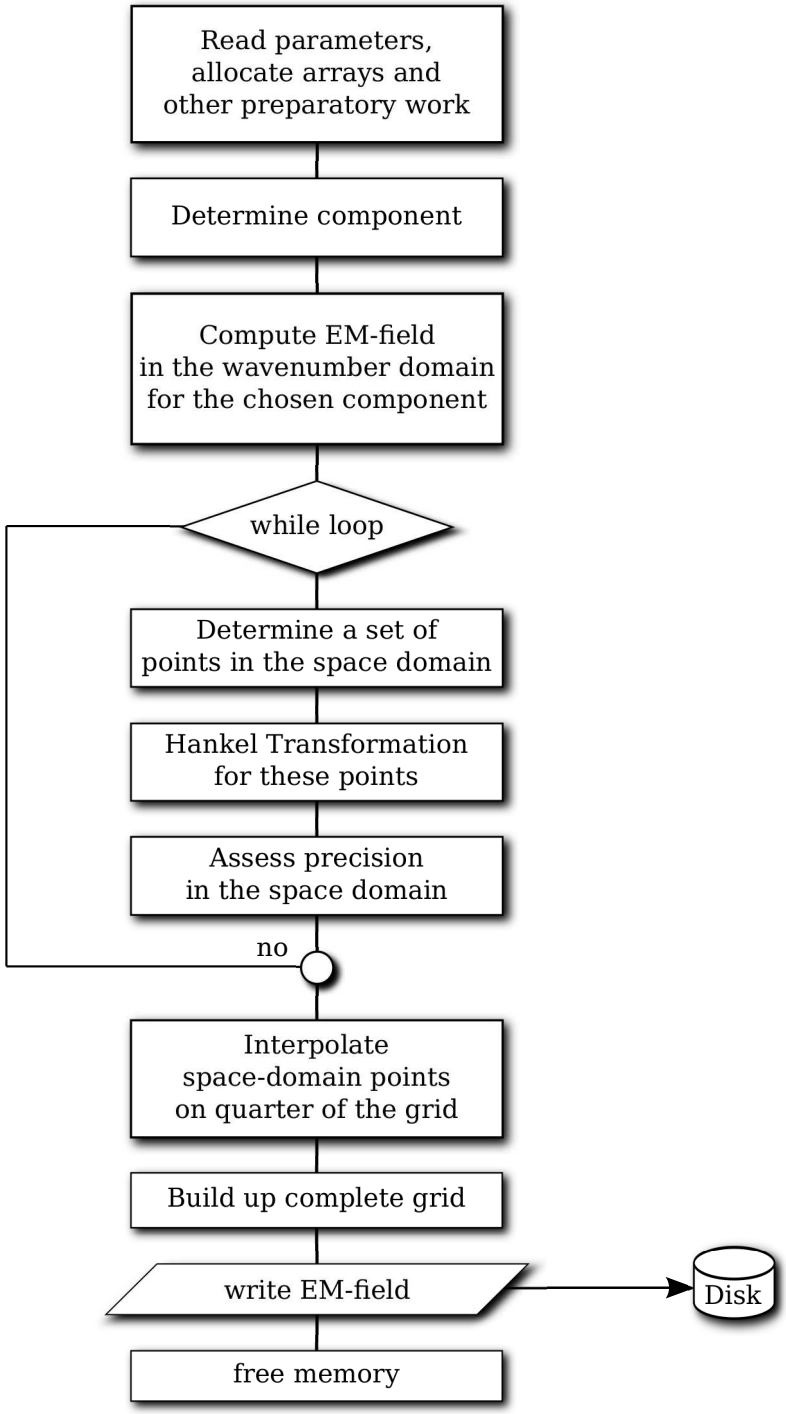


Figure 1: Structure of the EMmod program.

In the next section the input parameters are explained and their related options and limitations discussed. Subsequently, we show how the data can be loaded and plotted in Matlab. Since special attention was given to design a fast algorithm to be used for example as a forward-solver in an inversion code, we also present run-times of the code for various options.

2 Input parameters

EMmod offers to the user to enter 27 different input parameters. These input parameters can be specified using a script. Below the script found in the file `simplemod.scr` is shown. This script is included in the distribution of which this manual is also part of. The input parameters specified in this script assign EMmod to model electromagnetic data at inline oriented electric receivers for an inline oriented electric source. The model consist of a halfspace of air, a 200 m thick water layer and an anisotropic halfspace of sediments, which is intersected by a 200 m thick reservoir at a depth of 1 km.

```
#!/bin/bash

emmod \
        freq=0.5 \
        file_out=simplemod_11.bin \
        writebin=1 \
        nx=4000 \
        ny=1500 \
        zsrc=150 \
        zrcv=200 \
        dx=7 \
        dy=10 \
        z=-1,0,200,1000,1200 \
        econdV=0,3,0.1,0.02,0.1 \
        econdH=0,3,1,0.02,1 \
        epermV=1,80,17,2.1,17 \
        epermH=1,80,17,2.1,17 \
        mpermV=1,1,1,1,1 \
        mpermH=1,1,1,1,1 \
        verbose=1 \
        component=11 \
        nd=1000 \
        startlogx=-6 \
        deltalogx=0.8 \
        nlogx=16 \
        kmax=0.628625 \
        c1=-0.00083333 \
        c2=-0.0029167 \
        maxpt=500 \
        dopchip=0 \
        xdirect=1
```

Subsequently, we will discuss all input parameters in order of appearance in this script.

2.1 freq

The parameter `freq` defines the frequency of the signal in Hz. If it is intended to compute time-domain data, one needs to compute several frequencies. This can be done with a loop in the script calling EMmod changing the parameter `freq` in each iteration. The subsequent conversion to time can for example be done by Fourier transformation if high frequencies are used (e.g. GPR) or by the diffusion expansion method (Tehrani et al., 2012) if low frequencies are used (e.g. CSEM). If this parameter is not specified, EMmod will assume `freq=0.5`.

2.2 file_out

The parameter `file_out` specifies the name of the output file in which the data will be stored. If the file is to be stored in a different directory than the one in which EMmod is running, an absolute or a relative path including the name of the output file can be specified here as well. The file is stored in a binary format. In the next section, we show how the file is organized and how the data can be read into Matlab.

2.3 writebin

This parameter can have the values 1 or 0. If `writebin` is set to 1, the output file will be written in binary format. The Matlab scripts described in section 3 can be used for loading and plotting the binary data in Matlab. On the other hand, if `writebin` is set to 0, the output file will be written in ASCII format. If this parameter is not specified, EMmod will assume `writebin=1`.

2.4 nx and ny

The two parameters `nx` and `ny` specify the dimensions of the grid on which the electromagnetic field is computed. Thereby specifies `nx` the amount of datapoints in the x-direction (inline direction) and `ny` specifies the amount of datapoints in the y-direction (crossline direction). Note that values for `nx` and `ny` need to be even. If an uneven number is entered, the program will add 1 to make it even. Thus, the smallest grid that can be used is a grid of two by two elements. Note also that the coordinate vectors are symmetric with respect to the zero coordinate with one more datapoint on the negative axis. If an axis has only two datapoints, they will be at minus `dx` (or minus `dy`) and 0.

2.5 zsrc and zrcv

The two parameters `zsrc` and `zrcv` specify the depth in meters of the source and the receivers, respectively. The smaller the vertical seperation between the two, the broader the bandwidth of the direct field. If the source and the receivers are at the same depth, the bandwidth of the direct field is infinite. Solving the integral of the Hankel transformation numerically becomes then impossible. Thus, if the same depth for the source and the receivers is chosen, it is crucial to compute the direct field in the space domain. This can be done by setting the option `xdirect=1`. If the source and the receivers are at the same depth and also at an interface, not only the direct field but also the specular reflection of that interface have an infinite bandwidth. Since there is no option implemented to compute the specular reflection differently, in that case a small vertical offset between the source and the receivers has to be introduced. Consider also to increase the band of wavenumbers used for the computation by increasing the parameters `kmax` and `nd`.

2.6 dx and dy

The two parameters `dx` and `dy` define the sampling in meters in x-direction and y-direction, respectively. Note that the two parameters do not need to be equal.

2.7 z, econdV and econdH, epermV and epermH, mpermV and mpermH

These seven parameters are all arrays and are required to have the same amount of elements. Otherwise, EMmod will produce an error and stop. The parameter `z` contains the depth of the layer-interfaces of the medium in meters. Note that the very first value of the `z`-array will be disregarded. We recommend to just put the value -1 there. Note also, that if a source or a receiver depth coincides with an interface, the corresponding source or receiver will be considered above the interface. The two parameters `econdV` and `econdH` contain the vertical and horizontal electric conductivity for each layer in Siemens per meter. Similarly, `epermV` and `epermH` specify the vertical and horizontal electric permittivity relative to the value of free space, which is approximately $8.854 \cdot 10^{-12}$ Farads per meter

(Keller, 1988). Eventually, `mpermV` and `mpermH` specify the vertical and horizontal magnetic permeability relative to the value of free space, which is $4\pi \cdot 10^{-7}$ Henries per meter (Keller, 1988). The last four quantities are unitless because they are relative quantities.

2.8 verbose

This parameter can have the values 1 or 0. If `verbose` is set to 1, messages and warnings will be displayed. Accordingly, no messages and no warnings will be displayed if `verbose` is set to 0. If this parameter is not specified, EMmod will assume `verbose=0`.

2.9 component

The parameter `component` specifies the type and the orientation of both, the receivers and the source. This parameter consists of two numbers. The first one refers to the receiver while the second one refers to the source. The values 1, 2 and 3 represent an electric dipole antenna oriented in the inline-direction, the crossline-direction or the vertical direction, respectively. The values 4, 5 and 6 represent a magnetic dipole antenna oriented in the inline-direction, the crossline-direction or the vertical direction, respectively. Options 77 and 88 call the routine to compute the transverse magnetic (TM-mode) and the transverse electric (TE-mode) reflection response, respectively. These reflection responses form the "ground truth" for an electromagnetic interferometry experiment. All combinations except 77 and 88 are listed in table 1. Entry of any other combination will produce a message (if `verbose=1`) saying that the chosen component is not implemented. If this parameter is not specified, EMmod will assume `component=11`.

	inline electric source	crossline electric source	vertical electric source	inline magnetic source	crossline magnetic source	vertical magnetic source
inline electric receiver	11	12	13	14	15	16
crossline electric receiver	21	22	23	24	25	26
vertical electric receiver	31	32	33	34	35	36
inline magnetic receiver	41	42	43	44	45	46
crossline magnetic receiver	51	52	53	54	55	56
vertical magnetic receiver	61	62	63	64	65	66

Table 1: Choices for the parameter component to compute the electromagnetic field. Inline refers to parallel to the x-direction, while crossline refers to parallel to the y-direction. Note, that the choices 77 and 88, which refer to the TM-mode and TE-mode reflection responses, are not represented in the table.

2.10 nd

The integral is solved using a 61-point Gauss-Kronrod integration routine. This algorithm can not compute the complete integral in one go. Therefore, the integral is divided into subdomains. The parameter `nd` specifies how many subdomains. If this parameter is not specified, EMmod will assume `nd=1000`.

2.11 startlogx, deltalogx and nlogx

These three parameters define the coordinates in the space domain for which the Hankel transformation is carried out in the first iteration of the while loop. Thereby, `startlogx` specifies the logarithm of the coordinate of the first integration point greater than zero. The default value, which will be chosen if nothing is entered, is -6. This means, that the first coordinate is 10^{-6} meters. The parameter `deltalogx` specifies the logarithmic sampling rate. The default value is 0.025. Finally, the parameter

nlogx defines the amount of points of this coordinate vector. The default value is 512. It should be ensured, that the largest point of this coordinate vector is larger than the largest radial distance in the grid specified with the parameters dx, nx, dy and ny.

2.12 kmax

This parameter defines the upper boundary of the integral of the Hankel transformation. If the electromagnetic field in the wavenumber domain is large also at large wavenumbers, for example if the vertical distance between the source and the receivers is small or if the source and the receivers are close to an interface, this integration boundary has to be shifted to larger wavenumbers. It is crucial to also increase accordingly the parameter nd. Naturally, if the complete integral spreads over a larger amount of wavenumbers, it needs to be divided into a larger amount of subdomains. The default value is kmax=0.628625 m^{-1}, which is twice the Nyquist wavenumber for a spatial sampling rate of 10 m.

2.13 c1 and c2

These two parameters define the maximum allowed error for the optimization process searching for the points in the space domain for which the Hankel transformation has to be evaluated. These two parameters can be computed using the following Matlab script:

```
% determines the function used for the error bar in emmod
clear all; close all; clc;

xdata = [-6 -10.5 -15]; % logarithm of base 10 of the field-amplitude
ydata = [0.0025 0.005 0.01]; % maximum allowed error
eps = 10^(-6); % Stabilization parameter for the line-fitting process
fs = 18; % Fontsize
lw = 2; % Linewidth

% Find the two parameters of the linear regression
A(:,1) = xdata.';
A(:,2) = ones(size(xdata.'));
c = (A'*A+eps*eye(2,2))\A'*ydata.';

% Compute the function based on the two parameters previously determined
xvec = linspace(min(xdata),max(xdata),100);
yvec = c(1).*xvec+c(2);

figure;
plot(xdata,ydata*100,'ok','Linewidth',lw);
hold on;
plot(xvec,yvec*100,'k','Linewidth',lw);
hold off;
grid on
xlabel('log_{10} of Field-Amplitude','Fontsize',fs)
ylabel('Largest allowed error [%]','Fontsize',fs)
if sign(c(2)) == 1
    tempstring = '+';
else
    tempstring = '-';
```

```
end
c2string = [tempstring,' ',num2str(abs(c(2)))];
title(['y = ',num2str(c(1)),'x ',c2string],'Fontsize',fs)
legend('datapoints','determined function')
set(gca,'Fontsize',fs)
print('-depsc2','error_model.eps')
```

The array `xdata` of this Matlab script defines the logarithmic amplitude of the electromagnetic field for which the maximum allowed error is specified in the array `ydata`. The Matlab script fits a line through the specified points. The EMmod parameters `c1` and `c2` define the slope and the offset of this linear function, respectively. Figure 2 shows the plot produced by the Matlab script printed above. The circles are the entered points with the required precision and the red line is the function found to fit these plots. The equation with parameters `c1` and `c2` is given in the header of the plot. The idea behind the choice made here for the precision is, that a smaller relative error is required if the amplitude of the electromagnetic field is large and a larger relative error is allowed if the amplitude of the electromagnetic field is small. For a constant error, `c1` is set to 0.0 and `c2` to the desired error. For example for a constant error of 5%, `c2` is set to 0.005. If the error is chosen large, less points in the space domain are required and, therefore, the algorithm runs faster, but the resulting electromagnetic field is not so precise. If the error is chosen small, a very precise result can be expected. However, it is also possible, that the desired accuracy can not be achieved and the algorithm iterates until the maximum amount of points specified with the parameter `maxpt` is reached. If no value is specified by the user, `c1` is set to 0.0 and `c1` is set to 0.01.

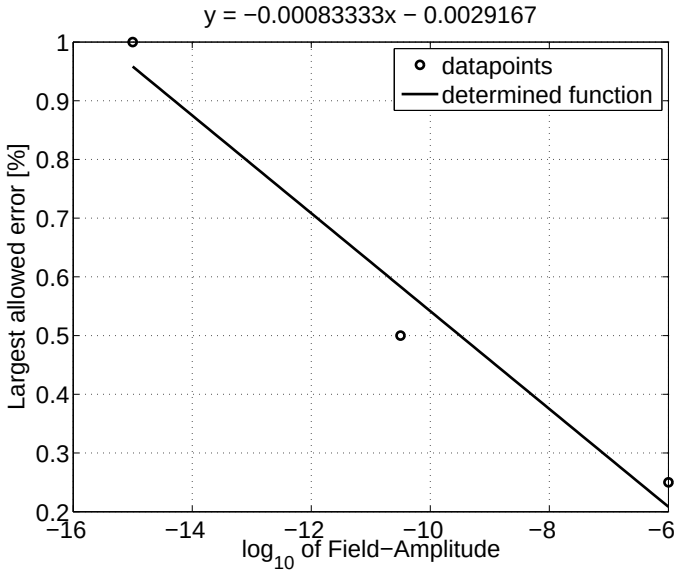


Figure 2: Error model: the circles are the datapoints for the required precision and the line is the determined function. The equation with parameters c1 and c2 is given above the figure.

2.14 maxpt

Under certain conditions the algorithm does not stop requiring more and more points in the space domain. This can for example be the case for components with a vertical electric antenna involved. These components decay typically at smaller offsets into the numeric noise. If that is the case, the algorithm will only fit numeric noise and will, therefore, never reach the required precision in that area. This can also happen if the precision parameters `c1` and `c2` are chosen to strictly. For these cases, the

user can define a maximum amount of points for which the Hankel transformation is computed. If no value is specified by the user, `maxpt` is set to 500.

2.15 dopchip

During the regridding of the Hankel transformed line of data in the space domain to a Cartesian coordinate grid, either pchip interpolation or linear interpolation is applied. If `dopchip` is set to 1, pchip interpolation is applied. If it is set to 0, linear interpolation is applied. Pchip interpolation requires less points in the space domain and is therefore faster. However, the interpolation is not everywhere so accurate. Therefore, we recommend linear interpolation for the best results. The default parameter is accordingly set to `dopchip=0`.

2.16 xdirect

The direct field from the source to the receiver can either be computed in the wavenumber domain and numerically Hankel transformed to the space domain or it can be computed directly in the space domain. If it is computed in the wavenumber domain, the code is slightly faster, but it is then impossible to put the source and the receivers at the same depth. If that is the case or if the vertical distance between the source and the receivers is small, which requires a larger wavenumber spectrum to be integrated, it is recommend to compute the direct field in the space domain. Note, if both, the source and the receivers, are close to the same interface, it is still necessary to chose a large wavenumber spectrum for the integration even if the direct field is computed in the space domain, because the specular reflection features a high wavenumber content. `xdirect=1` computes the direct field in the space domain, `xdirect=0` computes it in the wavenumber domain. The default parameter is 1. It has been observed, that computing the direct field in the space domain can lead to artifacts in the final output, because the direct field is computed on a different precision level as the rest of the electromagnetic field.

3 Load and plot output in Matlab

The binary output file generated by EMmod is a file without any header. The file contains 2 times `nx` times `ny` 64-bit real numbers with little-ending byte ordering. The factor 2 is because the electromagnetic field in the frequency-space domain is complex. Thus, for each value, two real numbers are required. The following script can be used to load and plot the binary EMmod output in Matlab:

```
% load and plots emmod output that has nx != ny
clear all; close all; clc; tic;

doabs = 1; % Plot absolute value (1) or no (0)
doang = 1; % Plot phase (1) or no (0)
folder = '../emmod_v9/';
filename = 'simplemod_11.bin';
xsize = 4000; % number of points in x-direction
ysize = 1500; % number of points in y-direction
dx = 7; % sampling in x-direction
dy = 10; % smapling in y-direction
fs = 18; % Fontsize

fprintf(['Load file ',filename,'...'])
[data,xvec,yvec] = loademmod_varsize([folder,filename],xsize,ysize,dx,dy);
data = data.';
fprintf('done\n')
```

```
if doabs == 1
    figure;
    imagesc(xvec/1000,yvec/1000,log10(abs(data)));
    caxis([-16 -7]);
    colorbar
    axis image
    xlabel('offset [km]','Fontsize',fs)
    ylabel('offset [km]','Fontsize',fs)
    title(['Amplitude of ',filename],'Fontsize',fs,'interpret','none')
    set(gca,'Fontsize',fs)
    print('-depsc2','amplitude.eps')
end

if doang == 1
    figure;
    imagesc(xvec/1000,yvec/1000,angle(data));
    caxis([-pi pi]);
    colorbar
    axis image
    xlabel('offset [km]','Fontsize',fs)
    ylabel('offset [km]','Fontsize',fs)
    title(['Phase of ',filename],'Fontsize',fs,'interpret','none')
    set(gca,'Fontsize',fs)
    print('-depsc2','phase.eps')
end

toc
```

Note that this Matlab script loads the data computed with the script containing the input parameters for EMmod shown earlier. This Matlab script calls the function `loademmod_varsize`, which is the following:

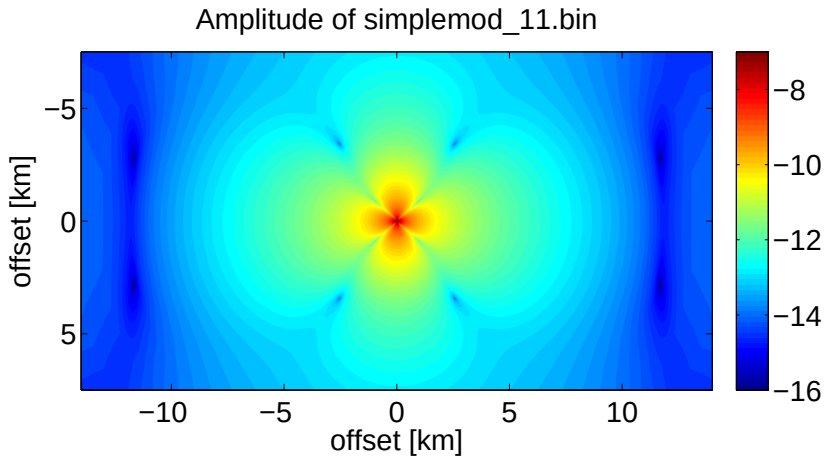
```
function [data,xvec,yvec] = loademmod_varsize(filename,xsize,ysize,dx,dy)
% LOADEMMOD_VARSIZE loads data created with emmod and the corresponding
% coordinate vectors
%
% Input arguments:
% - filename: path and name of the file to be loaded
% - xsize: number of points in x-direction
% - ysize: number of points in y-direction
% - dx: spacing in x-direction
% - dy: spacing in y-direction
%
% Output argument:
% - data: matrix containing the data
% - xvec: vector containing the coordinates in the x-direction
% - yvec: vector containing the coordinates in the y-direction
%
```

```
% Usage:
% [data,xvec,yvec] = loademmod(filename,xsize,ysize,dx,dy)

fid = fopen(filename,'r');

% Check if the filesize matches the number of elements specified
status = fseek(fid,0,'eof');
filesize = ftell(fid);
nel = round(filesize/8/2);
if nel~=xsize*ysize
    error('xsize and ysize are not specified properly.')
end
status = fseek(fid,0,'bof');

% Load the data
temp = fread(fid,2*xsize*ysize,'float64',0,'ieee-le');
fclose(fid);
temp2 = reshape(temp,[2,xsize,ysize]);
data = complex(squeeze(temp2(1,:,:)),squeeze(temp2(2,:,:)));
xvec = linspace(-xsize/2,xsize/2-1,xsize)*dx;
yvec = linspace(-ysize/2,ysize/2-1,ysize)*dy;
```

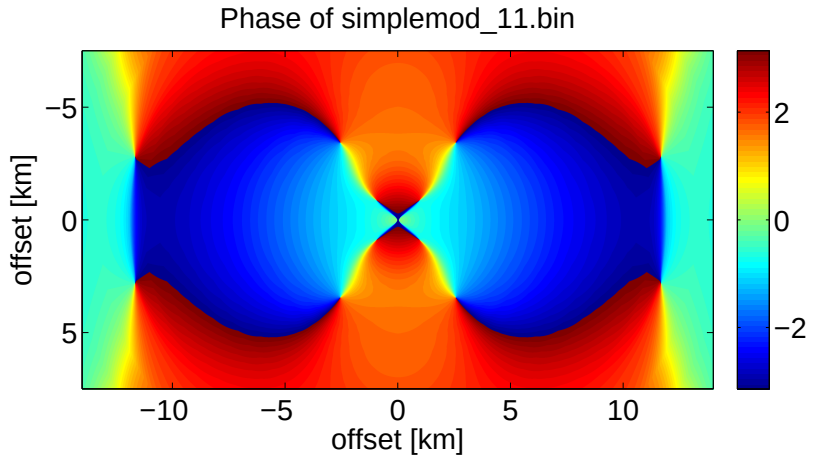
The figures produced by this Matlab script are shown in Figure 3.

4 Run-times

To demonstrate the speed of EMmod the script computing the simple model, which is shown above, was run with some variation of input parameters using one out of 32 AMD Opteron 6128 Processors with a clock rate of 2 GHz of our cluster. The script was run for two different components: 11 (inline electric receivers, inline electric source) and 13 (inline electric receivers, vertical electric source). These two components were chosen because they are fundamentally different with respect to their run-times. All components involving a vertical antenna will have similar run-times as the 13 component and all other components featuring only horizontal antennas will have similar run-times as the 11 component. While the algorithm is relatively soon satisfied for the 11 component concerning the points necessary for which the Hankel transformation has to be carried out, it is never satisfied for the 13 component. The latter is the case, because the 13 component drops at smaller offsets into the numerical noise than the 11 component. Points in the numerical noise can not be fitted. Therefore, the algorithm keeps adding new points until the maximum amount of points (`maxpt = 500`) is reached. Both components have been run for a grid of 6 million receivers (4000 in the inline direction times 1500 in the crossline direction) and a grid of 1.5 million receivers (2000 in the inline direction times 750 in the crossline direction). These are of course unrealistically huge amounts of receivers, but the result is a visually appealing picture. Table 2 compares run-times for linear interpolation (`dopchip = 0`), for pchip interpolation (`dopchip = 1`), for the direct field computed in the wavenumber domain (`xdirect = 0`) and for the direct field computed in the space domain (`xdirect = 1`).
Generally, the algorithm is faster when the direct field is computed in the wavenumber domain. It can also be seen from Table 2 that pchip interpolation is faster, because the integral needs to be evaluated fewer times. However, as stated earlier, pchip interpolation tends to be less accurate than linear interpolation. If the maximum amount of points `maxpt` is reached, as is the case for this example for the 13 component, pchip looses its advantage with respect to run-time, because more or less the same amounts of points are computed as for linear interpolation. Computing only a quarter of the points in



(a) Amplitude



(b) Phase

Figure 3: Amplitude and Phase of the EMmod output computed with the input parameters specified in simplemod.scr.

		4000 times 1500 receivers		2000 times 750 receivers	
		dopchip = 0	dopchip = 1	dopchip = 0	dopchip = 1
component=11	xdirect = 0	9.5	4.1	9.1	3.5
	xdirect = 1	13.8	8.3	10	4.6
component=13	xdirect = 0	13.5	13.1	13.1	12.6
	xdirect = 1	13.8	14.6	12	12.8

Table 2: Run-times in seconds for variations of the input script shown above. Each variation was run five times. The value given here is the average.

the space domain (1.5 million instead of 6 million) does not result in a quarter of the run-time. The reason is, that the integral still needs to be evaluated for the same amount of points in the space domain. Only the regridding steps are faster.

References

Hunziker, J., E. Slob, Y. Fan, R. Snieder, and K. Wapenaar, 2013, Electromagnetic interferometry in wavenumber and space domains in a layered earth: Geophysics, **78**, E137–E148.

Hunziker, J., E. Slob, J. W. Thorbecke, and F. C. Schoemaker, 2014, The electromagnetic response in

a layered vti medium: A new look at an old problem: Geophysics (under review).
Keller, G. V., 1988, Rock and mineral properties. In Nabighian, M. (Ed) Electromagnetic Methods in Applied Geophysics, Vol 1: Soc. Explor. Geophys., Tulsa.
Piessens, R., E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner, 1983, Quadpack, a subroutine package for automatic integration: Springer-Verlag.
Tehrani, A., E. Slob, and W. Mulder, 2012, Quasi-analytical method for frequency-to-time conversion in CSEM applications: Geophysics, **77**, E357–E363.